Addendum to: "A new numerical method for obtaining gluon distribution functions $G(x, Q^2) = xg(x, Q^2)$, from the proton structure function $F_2^{\gamma p}(x, Q^2)$."

Martin M. Block

Department of Physics and Astronomy, Northwestern University, Evanston, IL 60208

(Dated: April 22, 2010)

In a recent Letter entitled "A new numerical method for obtaining gluon distribution functions $G(x, Q^2) = xg(x, Q^2)$, from the proton structure function $F_2^{\gamma p}(x, Q^2)$" [1], we derived an accurate and fast algorithm for numerically inverting Laplace transforms, which we used in obtaining gluon distributions from the proton structure function $F_2^{\gamma p}(x, Q^2)$. We inverted the function $g(s)$, where s is the variable in Laplace space, to $G(v)$, where v is the variable in ordinary space. Since publication, we have discovered that the algorithm does not work if $g(s) \to 0$ less rapidly than $1/s$, as $s \to \infty$. Although we require that $g(s) \to 0$ as $s \to \infty$, it can approach 0 as $\frac{1}{s^\beta}$, with $0 < \beta < 1$, and still be a proper Laplace transform. In this note, we derive a new numerical algorithm for just such cases, and test it for $g(s) = \frac{\sqrt{\pi}}{\sqrt{s}}$, the Laplace transform of $\frac{1}{\sqrt{v}}$.

I. INTRODUCTION

In an earlier note [1] we developed an algorithm to numerically invert Laplace transforms, in order to find an analytic solution for gluon distributions. We used a global parameterization of the proton structure function, $F_2^{\gamma p}(x, Q^2)$ and a LO (leading-order) evolution equation for $F_2^{\gamma p}$. Subsequently, going to NLO (next-to-leading order) in the strong coupling constant $\alpha_s(Q^2)$, we have discovered the algorithm failed badly. Detailed investigation showed that the cause of the problem was that for this $g(s)$–whose Laplace transform was our desired NLO gluon distribution $G(v)$—it went to 0 *slower* than $1/s$ as $s \to \infty$, where s is the Laplace space variable. The purpose of this note is to derive a new algorithm for such cases, not covered in [1], which can be modeled by Laplace transforms of the type

$$g(s) = \frac{1}{s^\beta}, \qquad 0 < \beta < 1. \tag{1}$$

To illustrate the new method, we will numerically invert $g(s) = \frac{\sqrt{\pi}}{\sqrt{s}}$, the Laplace transform of $\frac{1}{\sqrt{v}}$, and test its accuracy, as well as show the inadequacy of the original algorithm for such a case.

II. NUMERICAL INVERSION OF LAPLACE TRANSFORMS

Let $g(s)$ be the Laplace transform of $G(v)$. The Bromwich inversion formula for $G(v)$ is given by

$$G(v) \equiv \mathcal{L}^{-1}[g(s); v] = \frac{1}{2\pi i} \int_{c-i\infty}^{c+i\infty} g(s) e^{vs} \, ds, \tag{2}$$

where the real constant c is to the right of all singularities of $g(s)$. Making an appropriate coordinate translation in s so that $c = 0$, we can write

$$G(v) \equiv \mathcal{L}^{-1}[g(s); v] = \frac{1}{2\pi i} \int_{-i\infty}^{+i\infty} g(s) e^{vs} \, ds. \tag{3}$$

Our goal is to *numerically* solve Eq. (3). The inverse Laplace transform is essentially determined by the behavior of $g(s)$ near its singularities, and thus is an ill-conditioned or ill-posed numerical problem. We suggest in this note a new algorithm that takes advantage of very fast, arbitrarily high precision complex number arithmetic that is possible today in programs like *Mathematica* [2], making the inversion problem numerically tractable.

First, we introduce a new complex variable $z \equiv vs$ and rewrite Eq. (3) as

$$G(v) = \frac{1}{2\pi i v} \int_{-i\infty}^{+i\infty} g\left(\frac{z}{v}\right) e^z \, dz. \tag{4}$$

In our earlier paper [1], we proceeded to make a rational approximation to the exponential e^z, under the tacit assumption that $g(s)$ went to 0 sufficiently rapidly as s went to ∞. This is *not* a valid assumption if $g(s)$ goes to 0 as

$1/x^\beta$, $0 < \beta < 1$. In this case, we will rewrite Eq. (4) as

$$G(v) = \frac{1}{2\pi i v} \int_{-i\infty}^{+i\infty} \left(\frac{g\left(\frac{z}{v}\right)}{z^2} \right) z^2 e^z \, dz \tag{5}$$

and make a rational approximation to $z^2 e^z$, using the Padé approximant

$$z^2 e^z \approx \text{Pade}[ze^z, 0, (2N - 3, 2N)], \tag{6}$$

where the numerator of the function Pade in Eq. (6) is a polynomial in z of order $2N - 3$ and whose denominator is a polynomial in z of order $2N$, and the expansion is around $z = 0$. Let α_i, $i = 1, 2, \ldots, 2N$ be the $2N$ complex roots of the denominator, i.e., its $2N$ complex poles, and let ω_i be the complex residues of the Padé. It can be shown that they have the following properties:

1. $\text{Re } \alpha_1 > 0$, so that its poles are all in the right-hand half of the complex plane.

2. N distinct complex conjugate pairs of the complex numbers (ω_i, α_i), such that the sum of the kth pair,

$$\frac{\omega_k}{z - \alpha_k} + \frac{\bar{\omega}_k}{z - \bar{\alpha}_k}, \tag{7}$$

is real for all real z.

3. The integrand vanishes *faster* than $1/R$ as $R \to \infty$ on the semi-circle of radius R that encloses the right hand half of the complex plane, since the approximation vanishes as $1/R$ and $g(s)$ that corresponds to a non-singular $G(v)$ also vanishes for $R \to \infty$.

Since g(z/v) must vanish for $z \to \infty$ and our approximation for $z^2 e^z$ in Eq. (6) *vanishes* for $z \to \infty$, we can form a closed contour C by completing our integration path of the modified Bromwich integral in Eq. (4) with an infinite half circle to the *right* half of the complex plane. As mentioned earlier, $g(z/v)$ has no singularities in this half of the complex-Z plane. It is important to note that this contour is a *clockwise* path around the poles of Eq. (6), which come from our approximation to ze^z. What we need is the negative of it, i.e., the contour $-C$ which is counterclockwise, so that the poles are to our left as we traverse the contour $-C$. Therefore, we rewrite Eq. (4) as

$$\begin{aligned}
G(v) &\approx \frac{1}{2\pi i v} \oint_C \frac{g\left(\frac{z}{v}\right)}{z^2} \text{Pade}[z^2 e^z, 0, (2M - 3, 2M)] \\
&= -\frac{1}{2\pi i v} \oint_{-C} \frac{g\left(\frac{z}{v}\right)}{z^2} \text{Pade}[z^2 e^z, 0, (2M - 3, 2M)] \\
&= -\frac{2}{v} \sum_{i=1}^{N} \text{Re}\left[\omega_i \frac{g(\alpha_i/v)}{\alpha_i^2} \right].
\end{aligned} \tag{8}$$

To obtain Eq. (8), the final approximation to $G(v)$, we used Cauchy's theorem to equate the closed contour integral around the path $-C$ to $2\pi i$ times the sum of the (complex) residues of the poles. Since the contour $-C$ restricts us to the right-hand half of the complex plane, no poles of $g(z/v)$ were enclosed, but only the $2N$ poles α_i of the Padé approximant of $z^2 e^z$. Using the properties cited above of the complex conjugate pairs—(ω_i, α_i) and $(\bar{\omega}_i, \bar{\alpha}_i)$—after taking only their real part and multiplying by 2–we have simultaneously insured that $G(v)$ is real , yet only have had to sum over half of the residues. The above scheme now works for the slowly converging case of $g(s) = 1/s^\beta$, where $0 < \beta < 1$, because we are *dividing* it by z^2 to make the quantity that we are inverting , i.e., $g(z/v)/z$, go to 0 *more rapidly* than $1/s$ as $s \to 0$.

Two properties of Eq. (8) are worth emphasizing here:

1. The $4N$ coefficients (α_i, ω_i) are complex constants that are independent of v, only depending on the value of $2N$ used for the approximation, so that for a given $2N$, they need to be evaluated only once—in essence, they can be tabulated and stored for later use.

2. The real parts of the residues ω_i alternate in sign and are *exceedingly* large—even for relatively modest $2N$, making round-off a potentially serious problem. Thus, exceedingly high precision complex arithmetic is called for, often requiring 60 or more digits. However, this is not a serious problem—either in speed or complexity of execution—for an algorithm written in a program such as *Mathematica* [2].

A concise inversion algorithm in *Mathematica* that implements Eq. (8) is given in Appendix A.

We will now test the accuracy of our numerical Laplace inversion algorithm by using $g(s) = \frac{\sqrt{\pi}}{\sqrt{s}}$, the Laplace transform of $G(v) = \sqrt{v}$.

III.   COMPARISON OF EXACT SOLUTION AND NUMERICAL LAPLACE INVERSION RESULTS

To illustrate our inversion routine for slowly converging Laplace transforms, we plot in Fig. 1 the numerical inversion of $g(s) = \frac{\sqrt{\pi}}{\sqrt{s}}$, the Laplace transform of $G(v) = \frac{1}{\sqrt{v}}$. The solid curve is the function $G(v) = \frac{1}{\sqrt{v}}$, and the (blue) dots are a result of using the algorithm in Appendix A, using Mathematica [2], with $2N = 20$. As seen, the agreement is excellent. Also shown in Fig. 1 are the (red) squares, which are the result of our using our earlier algorithm [1] for the calculation of the inverse Laplace transform. Clearly, this is a disaster. As detailed in our earlier work [1], the original algorithm was *exact* if the function in v space was a polynomial of degree $\leq 4N - 1$, so that it is a very powerful tool when used properly, i.e., with a $g(s)$ that goes to 0 as fast or faster than $1/s$ as $s \to \infty$. However, for $g(s)$'s that don't vanish this rapidly, that algorithm is deficient and must be replaced by the new algorithm given in this note.

As a practical matter, if we have a $g(s)$ to invert that is purely numerical and whose behavior at ∞ is completely unknown (except for the requirement that it *must* vanish at ∞), we must test its convergence properties by trying *both* algorithms. If they agree reasonably numerically, we should continue using the more powerful algorithm of Ref. [1]. It they disagree completely, as is the case for the squares in Fig. 1, then one should use the new algorithm developed in this note.

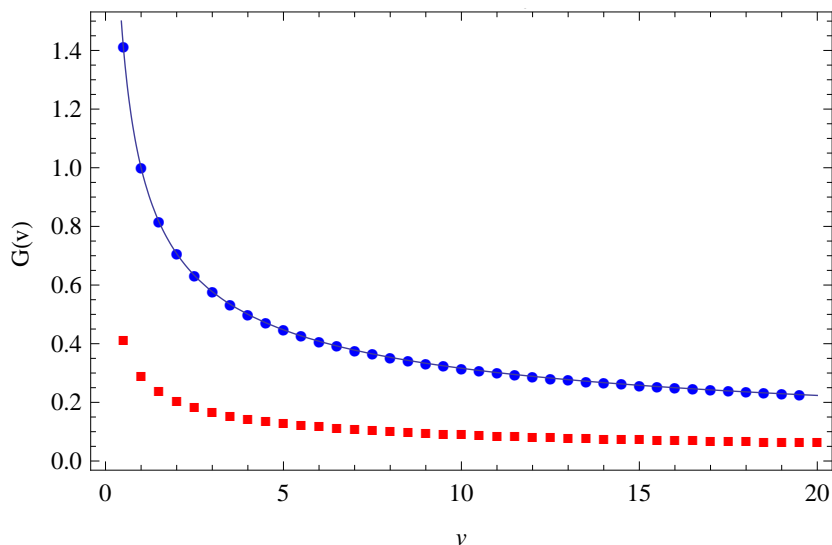We will discuss in Appendix A the inherent accuracy of the algorithm used to get the dots in Fig. 1. As noted in

FIG. 1: A plot of $G(v) = \frac{1}{\sqrt{v}}$ vs. v. The solid curve is $G(v)$. The (blue) dots are the approximate values calculated from Eq. (8), using $2N =20$ and $g(s) = \frac{\sqrt{\pi}}{\sqrt{s}}$; the (red) squares are the result of using $2N = 20$ in the algorithm described in Ref. [1], which should *only* be used for Laplace transforms that go to 0 at ∞ as fast, or faster than $1/s$.

[1], our inversion routine clearly has a wide variety of additional applications in solving both integral and differential equations.

IV.   CONCLUSIONS

We have achieved arbitrarily high accuracy in numerically inverting Laplace transforms of the type $g(s) = \frac{1}{s^\beta}$, $0 < \beta < 1$. As an example, we show in Fig. 1 a comparison of the solution $G(v) = \frac{1}{\sqrt{v}}$, i.e., the inverse Laplace transform of $g(s) = \frac{\sqrt{\pi}}{\sqrt{s}}$, with the exact answer, and demonstrate the algorithm's inherent accuracy in Fig. 2, showing that it is a power law in $\frac{1}{2N}$, the reciprocal of the number of terms used in the expansion, and thus, of arbitrary accuracy. Because the assumed $g(s)$ go to 0 very slowly as $s \to \infty$, the algorithm that we previously developed in Ref. [1] does *not* work here, and we discuss when and when not to use it. We are currently using our new algorithm (this communication) to analytic decoupled solutions for LO singlet structure functions and gluons [3] , as well as for obtaining NLO gluon distributions.

Appendix A: *Mathematica* Laplace Inversion Algorithm

```
NInverseLaplaceTransformBlock2[g_,s_,v_,twoN_,prec_]:=Module[
   {Omega,Alpha,M,p,den,r,num},
   (M=2*Ceiling[twoN/2];If[M<6,M=6];p=PadeApproximant[ z^2*Exp[z],{z,0,{M-3 ,M}}];
   den=Denominator[p];r=Roots[den==0,z]; Alpha=Table[r[[i,2]],{i,1,M}];
   num=Numerator[p];hospital=num/D[den,z];
   Omega=SetPrecision[Table[hospital/.z->Alpha[[i]],{i,1,M,2}],prec+50];
   Alpha=SetPrecision[Table[Alpha[[i]],{i,1,M,2}],prec]);
   SetPrecision[-(2/v)Sum[Re[Omega[[i]] (g/(v*s)^2)/.s->Alpha[[i]]/v],{i,1,M/2}],prec]
                                                                                  ]
```

In the above algorithm, $g = g(s)$, $s = s$, $v = v$, `twoN`=$2N$ in Eq. (8), and `prec` = desired precision of calculation. Typical values are `twoN` $= 10$ and `prec` $= 70$. The algorithm, which is quite fast, returns the numerical value of $G(v)$.

The algorithm first insures that M=`twoN` is *even* and ≥ 6. It then constructs p, the Padé approximant whose numerator is a polynomial in z of order $M - 3$ and denominator a polynomial in z of order M, whose expansion is around $z = 0$. It finds r, the complex roots of the denominator, which are α_i, the poles of Eq. (8). Using L'Hospital's rule, it finds the residue ω_i corresponding to the pole α_i. At this point, all of the mathematics is symbolic. It next finds every *other* pair of (α_i, ω_i) to the desired numerical accuracy; they come consecutively, i.e., $\alpha_1 = \bar{\alpha}_2$, $\omega_1 = \bar{\omega}_2$, $\alpha_3 = \bar{\alpha}_4$, $\omega_3 = \bar{\omega}_4$, etc. Finally, it takes the necessary sums, again to the desired numerical accuracy, but only over half of the interval $i = 1, 3, \ldots, N$, by taking only the real part and multiplying by 2.
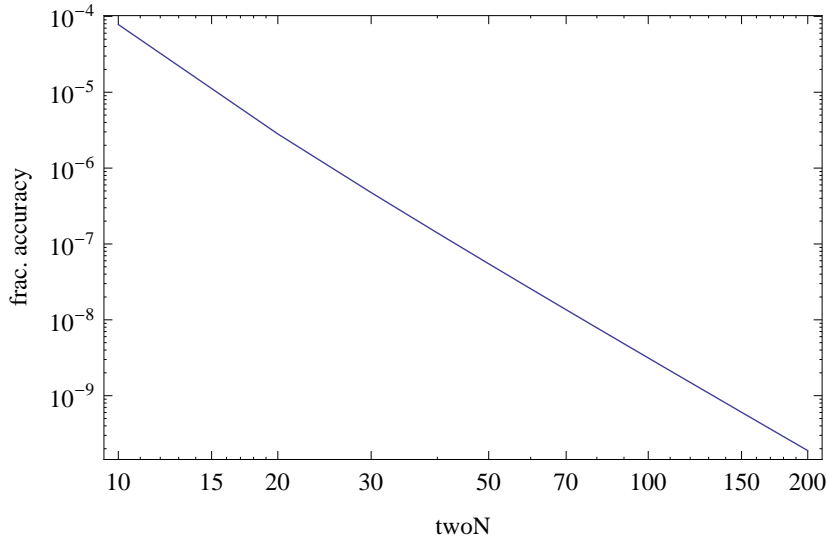

FIG. 2: A log-log plot of the fractional accuracy of the numerical Laplace inversion of the original function $G(v) = \frac{1}{\sqrt{v}}$ vs. twoN, with twoN=$2N$, the order of the rational approximation in Eq. (8).

If $g(s)$, the input to the algorithm, is an *analytic* relation *and* v is a pure number (from the point of view of *Mathematica*, 31/10 is a pure number, but 3.1 is *not*), then, for sufficiently high values of `prec`, you can achieve arbitrarily high accuracy. If we define the fractional accuracy as $1 - G(v)_{\text{numerical}}/G(v)_{\text{true}}$, numerical tests on many different functions shows that it goes to 0 for large `2N` as a power law in $1/2N$. We illustrate this in Fig. 2, where we show a log-log plot of the fractional accuracy vs. $2N$. The straight line shows that the fractional accuracy is a power law in $2N$, allowing one to achieve arbitrary accuracy in the inversion of $g(s) = \frac{\sqrt{\pi}}{\sqrt{s}}$. Further discussion of applications and numerical properties can be found in Ref. [1].

Acknowledgments

parts of this work were done.

_____

[1] M. M. Block, Eur. Phys. J. C **65**, 1 (2010).
[2] *Mathematica* 7, a computing program from Wolfram Research, Inc., Champaign, IL, USA, www.wolfram.com (2009).
[3] M. M. Block, L. Durand, P. Ha, and D. W. McKay, to be published, (2010), arXiv:1004.1440 [hep-ph].